PATENT APPLICATION
4233-4005

27123
PATENT TRADEMARK OFFICE

UNITED STATES PATENT APPLICATION

For

SYSTEM AND METHOD FOR SAVING AND MANAGING BROWSED DATA

Inventor(s):
ARUNA ROHRA SUDA
SURESH JEYACHANDRAN
PREM ANAND JOSEPH

Express Mail label No.: EF098974587US
Date of Deposit: August 24, 2001

Arun Chandra (Reg. No. 43,537)
MORGAN & FINNEGAN, LLP
345 PARK AVENUE
NEW YORK, N.Y. 10154-0053
(212) 415-8573
(212) 751-6849 (FACSIMILE)
WWW.MORGANFINNEGAN.COM

SYSTEM AND METHOD FOR SAVING AND MANAGING BROWSED DATA

CLAIM FOR PRIORITY

This application claims priority from Application No. 2001-107893, filed on May April 6th, 2001 in JAPAN.

FIELD OF THE INVENTION

The present invention relates to a system and a method for saving and managing acquired data that is, for example, browsed by an Internet browser.

BACKGROUND OF THE INVENTION

A conventional computer can access and display data in an Internet using a software called browser. The conventional computer also can open a file application and store data in a file.

As the Internet becomes the primary communication infrastructure, browser is becoming the normal desktop, not only for professional users but even for the layman and the need for web information management while browsing becoming a 'basic necessity'. The key issue for professional users thus, is the cultivation of 'information power' i.e., harnessing and effective use of the web. The rapidly growing and changing web information, the dynamically created context based information (Electronic Commerce receipts, search results, individually tailored insurance plans etc.), make the conventional management

techniques like book marking, printing and filing, ineffective. Therefore, the user has to perform complex operations to save the browsed data and manage the data stored in files.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a system and method to save data by a user using a simple operation.

According to one aspect, the present invention relates to a data processing system comprising a data acquisition means for acquiring data; determination means for determining whether or not a user requests to save the acquired data; indexing means for assigning a predetermined index to the data requested to save, without inputting any index; and saving means for saving the requested data with the assigned index in a predetermined storage unit.

According to another aspect, the present invention relates to a data processing method comprising a data acquisition step of acquiring data; a determination step of determining whether or not a user requests to save the acquired data; an indexing step of assigning a predetermined index to the data requested to save, without inputting any index; and a saving step of saving the requested data with the assigned index in a predetermined storage unit.

According to still another aspect, the present invention relates to a computer-executable program for controlling a computer to perform data processing, said program comprising codes for causing the computer to perform a data acquisition step of acquiring data; a determination step of determining whether or not a user requests to save the acquired data; an indexing step of

assigning a predetermined index to the data requested to save, without inputting any index; and a saving step of saving the requested data with the assigned index in a predetermined storage unit.

Other features and advantages of the present invention will be apparent from the following description taken in conjunction with the accompanying drawings, in which like reference characters designate the same or similar parts throughout the figures thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention.

Fig. 1 is a block diagram illustrating the hardware configuration according to an embodiment of the present invention.

Fig. 2 shows the functional block diagram of information processing system embodying the present invention.

Fig. 3 shows the functional block diagram of information browsing system embodying the present invention.

Fig. 4 is a flowchart of the main procedural steps of this embodiment.

Fig. 5 is a flowchart of INITIALIZE procedure.

Fig. 6 is a flowchart of MAIN-PROCESSOR.

Fig. 7 is a flowchart of the procedural steps of UserAction.

Fig. 8 is a flowchart of the procedural steps of CheckExisting().

Fig. 9 is a flowchart of the procedural steps of ShowSession().

Fig. 10 is a flowchart of the procedural steps of For Each Session, Sort KPTAction by Time.

Fig. 11 is a flowchart of the procedural steps of GetPreviousYearNodes.

Fig. 12 is a flowchart of the procedural steps of GetBeforeCurrMonthNodes.

Fig. 13 is a flowchart of the procedural steps of GetbeforeCurrWeekNodes.

Fig. 14 is a flowchart of the procedural steps of GetCurrWeekNodes.

Fig. 15 is a flowchart that procedural steps of ShowLinks().

Fig. 16 is a flowchart of the procedural steps of Sort by keywords.

Fig. 17 is a flowchart of the procedural steps of ShowTreeView.

Fig. 18 is a flowchart of the procedural steps of ProcessLeafAction.

Fig. 19 is a flowchart of the procedural steps of ProcessNodeAction.

Fig. 20 is a flowchart of the procedural steps of Sort by Domain.

Fig. 21 is a flowchart of the procedural steps of Sort by Organization.

Fig. 22 is a flowchart for processing the retrieve conditions and to fetch the appropriate information.

Fig. 23 is a flowchart for processing the manage conditions and to perform the appropriate processing.

Fig. 24 is a flowchart of the procedural steps to see if any system task needs to be executed.

Fig. 25 is a flowchart of the procedural steps of RemoveExpired().

Fig. 26 is a flowchart of the procedural steps of CheckUpdate().

Fig. 27 is a flowchart of the procedural steps of ExecuteAction().

Fig. 28 is a flowchart of the procedural steps of SaveContents().

Fig. 29 is a flowchart of the procedural steps of WebFetch().

Fig. 30 is a flowchart of the procedural steps of steps of SaveFileContents().

Fig. 31 is a flowchart of the procedural steps of FillKPTAction.

Fig. 32 is a flowchart of the procedural steps of TERMINATE.

Figs. 33 and 34 show examples of the knowledge structures in the knowledge base.

Figs. 35 and 36 show examples content of the knowledge base.

Fig. 37 shows an example content of the Domain knowledge base.

Fig. 38 shows example rules for determining the organization indices for a URL.

Fig. 39 shows example rules for determining the domain indices for a URL.

Fig. 40 shows an example User Interface of this embodiment.

Fig. 41 shows an example HTML text obtained from the browser.

Fig. 42 shows example modified HTML text.

Fig. 43 shows example User Interface, when a saved page is retrieved.

Fig. 44 shows an example User Interface of Session UI.

Fig. 45 shows an example User Interface of Get All Links UI sorted by Organizations.

Fig. 46 shows an example User Interface of Get All Links UI sorted by Domains.

Fig. 47 shows an example User Interface of Get All Links UI sorted by

Keywords.

Fig. 48 shows an example Retrieve or Find UI.

Fig. 49 shows an example Properties or Attributes UI of a retrieved page.

Fig. 50 shows an example Manage Data or Delete Pages UI.

Figs. 51 to 53 shows some examples of User Settings UI.

DETAILED DESCRIPTION

A preferred embodiment of the present invention will now be described in detail with reference to the accompanying drawings.

Fig.1 is a block diagram illustrating the hardware configuration according to an embodiment of the present invention. In this figure, A central processing unit (CPU) 101 is operative to perform operations for various processing and make a logical decision or the like and further controls each composing element connected to a bus 107.

A RAM 102 is used to temporarily store variables and intermediate data generated during the processing. A program from an external source may be loaded into the RAM 102. A ROM 103 is used to store programs, which correspond to individual flowcharts that will be described later and which are to be executed by the CPU 101, and fixed data.

A user uses a keyboard (KB) 104 for inputting data and an instruction. A mouse or other input devices may be used with the keyboard 104. Display 105 displays data and a hard disk drive (HDD) stores data of a database, a program, and the like.

The bus 107 is used to transfer an address signal indicating a composing

element to be controlled by the CPU 101, a control signal used for controlling each composing element and data to be exchanged between the composing equipment.

Fig. 2 shows the functional block diagram of information processing system embodying the present invention. Browser 201 is used for information browsing of the Web. Filing system 202 is for filing and managing files. Document Management System 203 is for managing documents. Information Management System 204 is for managing information other than documents. The inventive System interacts and acts as a controlling system as explained in detail in this embodiment to Browser 201, Filing System 202, Document Management System 203 and Information Management System 204. Knowledge Base Management 206, is the management of knowledge accessed/stored from/to the Database 206.

Fig. 3 shows the functional block diagram of information browsing system embodying the present invention. The information on the Internet 301 is browsed using multiple browsers 302, 303 simultaneously and as explained in this embodiment, the System 205 handles and processes them separately.

Fig. 4 is a flowchart of the main procedural steps of this embodiment. The following description is for an example system, which connects to the Internet and allows browsing and saving of the information. In step S401, initialization processes to connect to the Internet are executed. In step S402, main function processing browsing, saving, managing stored information etc. of this embodiment is performed. In step S403, terminate or clean-up processing is executed.

Fig. 5 is a flowchart of INITIALIZE procedure of step S401. In step S501 a check is made to determine if the browser needs to be instantiated or not. If browser is not instantiated, it is instantiated in step S502. In step S503, a new session is created. In step S504, the knowledge base is updated. The main UI of this embodiment is displayed in step S505 and the process ends.

Fig. 6 is a flowchart of MAIN-PROCESSOR of step S402. In step S601, a check is made to determine if the browser was instantiated or not. If so, a new session is created in step S602 and the process proceeds to step S605, wherein the knowledge base is updated. If not, a check is made in step S603 to determine if the browser was terminated or ended. If so, the session associated with the browser is ended and proceeds to step S605. If not, a check is made in step S606 to determine if an action was performed to end the system. If so, all the current tasks are terminated in step S607 and the process returns.

If not, a check is made in step S608 to determine if the user is navigating to a new URL. If so, a check is made in step S609 to confirm with the user that the current task should be terminated. If not, the process proceeds to step S610, where the navigation is aborted and the process continues to step S601. If the current task is to be ended in step S609, step S611 is executed wherein, the previous task is terminated and then a new task is created. In step S612, the knowledge structures KPTAction and KPTDocument are created.

In step S613, the URL and the keywords are obtained from the Browser. A check is made in step S614 to determine if the URL data already exists in the knowledge base. If so, all the existing data for the current URL is procured from the knowledge base in step S615 and moves to step S616, where a check is made

to determine if it is a RetrievedURL i.e., the user is trying to view the contents of an already stored page. If so, step S617 is executed to get the RetrieveUI message and control goes to S618. If URL data does not already exist in step S614, step S618 is executed to display the keywords, other acquired data from browser like the URL, page title etc... and other existing data if any from the knowledge base like validity period etc... and the process proceeds to step S601.

In step S608, if the user is not moving to a new URL, a check is made in step S619 to determine if any SystemTask ActL needs to be executed. If so, step S622 ExecuteAction(ActL) is executed and control moves to step S605 to update the knowledge base. If not, a check is made in step S620 to determine if any User Operation was performed. If not, step S605 is executed, otherwise in step S621, the HTML text is obtained from the browser and the KPTAction and KPTDocument structures created in step S612 are updated and ExecuteAction(ActL) for the UserAction is executed in step S622 and the process moves to step S605 to update the knowledge base.

For example, when the browser 201 displays an HTML page as shown in Fig. 41, the keywords embedded in the Meta Name tag i.e., Saora, Web Information Manager etc., which are not normally displayed by the browser are extracted and displayed in the keywords field, as shown in Fig. 40. As can be seen from this Figure, the User Interface of this embodiment appears within the browser. Thus this embodiment allows the user to do all his information gathering, processing and management while browsing, without ever leaving the browser. The contents can, not only be saved with a single mouse click but also assign keywords and keep track of them. The embodiment also automatically sorts

and organizes the web research results, which are easily accessible in time and content-sorted order or for search by keywords etc. Most importantly, there is no need to specify filenames, folder destinations etc. so that there is no wasted time filing, sorting, organizing and printing for offline reading.

As can be seen in Figure 40, this embodiment allows the user to save information from web pages and retrieve the stored information, without alluding to any files and folders; concepts normally associated with saving information on a computer. There is no need to specify a filename or a destination folder. This saves a lot of time spent in deciding for a filename and specifying the destination folder and after time passes, finally searching the whole hard disk(s) for the files one is looking for. There is not only no need to specify a filename, but the link or pages can be kept, in an One-Click operation i.e., by just clicking on either Save Link or Save Page. Also the same contents can be stored as many times as the user wishes, without ever specifying the filename, and still be able to retrieve them easily. This is especially useful, if the user wishes to keep track of the changes in the contents of the same web page.

Fig. 7 is a flowchart of the procedural steps of S620 UserAction. A check is first made in step S701 to set Act equal to the User operation performed by user and proceeds to step S702 to determine if Act is equal to NULL. If so, the process returns false. If it is not NULL, the process proceeds to step S703, a check is made to determine whether Act is Save. If Act is Save, in step S704, a check is made to determine if the information being saved already exists using CheckExisting(), which is explained in detail later and if so, proceeds to step S715, otherwise return false. If not, a check is made in step S705 to determine if

Act is Show Sessions. If so, ShowSession() as explained later is executed in step S706 and the process returns true. If not, a check is made in step S707 to determine if the Act is Show Links. If so, ShowLinks() as explained later is executed in step S708 and the process returns true. If not, a check is made in step S709 to determine whether Act is Retrieve. If Act is retrieve, the process proceeds to step S710 where RetrieveUI is displayed, example of which is shown in Fig. 48. If not, a check is made in step S711 to determine whether Act is Manage. If Act is manage, the process proceeds to step S712 where ManageUI is displayed, example of which is shown in Fig. 50. If not, a check is made in step S713 to determine whether Act is Show User Settings. If Act is User Settings, the process proceeds to step S714 where User SettingUI is displayed, example of which are shown in Figs. 51 to 53. In step S715, createKS() is executed to create the associated knowledge structures, which are explained in detail later and the process returns true.

Fig. 8 is a flowchart of the procedural steps of S704, CheckExisting() of this embodiment to check if the information already exists in the knowledge base or not. In step 801, the values of Keyword, Validity Range etc... are either obtained from the user or from the system settings. In step S802, a check is made to determine whether the URL already exists. If URL does not exist, the process proceeds to step S803 where ModifyStatus is set to saveAsNewAction and returns true. If URL exists, a check is made in step S804 to determine if the information needs to be over-written i.e., update or modify the previous information. This is done either by asking the user, whether he or she would like to overwrite the existing information, save as new, do not save or based on the user settings as

shown in Fig. 53. If so, in step S805, ModifyStatus is set to OverWriteExisting and the process returns true. If not, a check is made in step S806 to determine if the information needs to be saved as new, i.e., without modifying the existing information, save the current information as a separate entity. If so, ModifyStatus is set to saveAsNewAction and the process returns true. If not, the process returns false.

Fig. 9 is a flowchart of the procedural steps of S706, ShowSession of this embodiment. A check is made in step S901 to determine if all the information needs to be displayed. If not, the process proceeds to step S902, wherein based on either User settings and/or interaction and/or input and/or System settings, the filter or the constraint on the information to be displayed is obtained. In either case the process proceeds to step S903, wherein the relevant KPTAction and the associated KPTDocument are got from Knowledge Base. In step S904, KPTAction is sorted for each session by time as explained in detail later, in Fig. 10. In step S905, Session UI is displayed, an example of which is shown in Fig. 44 and the process returns.

Fig. 10 is a flowchart of the procedural steps of For Each Session, Sort KPTAction by Time, step S904 of this embodiment. Initially in step S1001, the CurrD is set to GetCurrentDate(), which the current date of the system. In step S1002, the MinD is set to GetMinimumDate(), which the minimum or earliest date for which information exists in the knowledge base. In step S1003, the NodeList is set to NULL. In step S1004, GetPreviousYearNodes(NodeList, CurrD, MinD) as detailed in Fig. 11 is executed. In step S1005, GetBeforeCurrMonthNodes(NodeList, CurrD) as detailed in Fig. 12 is executed.

In step S1006, GetBeforeCurrWeekNodes (NodeList, CurrD) as detailed in Fig. 13 is executed. In step S1007, GetCurrWeekNodes(NodeList, CurrD) as detailed in Fig. 14 is executed. In step S1008, ShowTreeView(NodeList, Type) with Type = Session is executed as detailed in Fig. 17 and the example results are shown in Fig. 44, wherein the Sessions in the NodeList are displayed in sorted order.

Fig. 11 is a flowchart of the procedural steps of GetPreviousYearNodes (NodeList, CurrD, MinD) of step S1004 of this embodiment. A check is made in step S1101 to determine if Year(CurrD) > Year (MinD) i.e., the year part of CurrD is greater than the year part of MinD. If not, the process returns. If so, Iyear is set to Year(MinD) i.e., year part of MinD in step S1102. In step S1103, a check is made to determine if Iyear < Year(CurrD) i.e., Iyear is less than the year part of CurrD. If not, the process returns. If so, a check is made in step S1104 to determine if Iyear is equal to Year(CurrD) – 1. If so, a node called 'Last year' is created and added to the appropriate place in NodeList in step S1105. If not, nodes for that year i.e., Iyear is created e.g., 1999 etc… and added to appropriate place in NodeList in step S1106. In either case, the step S1107 is executed, wherein nodes are created only for 'Months' e.g., January, June etc… for which data exists for the specified Iyear in the knowledge base and added to appropriate places in NodeList. In step S1108, nodes are created only for 'Days' e.g., 1, 3, 27 etc… for each of the above 'Months' for which data exists in the knowledge base and are added to appropriate places in NodeList. In step S1109, Iyear is incremented and the process continues to step S1103, till Iyear becomes greater than or equal to year part of CurrD, at which point the process returns.

Fig. 12 is a flowchart of the procedural steps of

GetBeforeCurrMonthNodes (NodeList, CurrD) of step S1005 of this embodiment. Initially, in step S1201, Imonth is set to 1. A check is made in step S1202 to determine if Month(CurrD) i.e., month part of CurrD is equal to Imonth. If so, the process returns. If not, a check is made in step S1203 to determine if the values of Imonth and Month(CurrD)-1 are equal. If so, step S1204 is executed in which, a node called 'Last month' is created and added to appropriate place in NodeList. If not, step S1205 is executed in which, the node with Month name e.g., Jan or Jun etc... is created and added to appropriate place in NodeList. In either case, after completion, step S1206 is executed, wherein nodes are created only for 'Days' e.g., 1, 3, 27 etc... for each of the above 'Months' for which data exists in the knowledge base and are added to appropriate places in NodeList. In step S1207, Imonth is incremented and the process continues to step S1202, till Imonth is equal to the month part of CurrD, at which point the process returns.

Fig. 13 is a flowchart of the procedural steps of GetbeforeCurrWeekNodes (NodeList, CurrD) of step S1006 of this embodiment. Initially, in step S1301, Iweek is set to 1. A check is made in step S1302 to determine if Week(CurrD) i.e., week part of CurrD is equal to Iweek. If so the process returns. If not, a check is made in step S1303 to determine if the values of Iweek and Week(CurrD-1) are equal. If so, step S1304 is executed in which, a node called 'Last Week' is created and added to appropriate place in NodeList. If not, step S1305 is executed in which , the node with Iweek is created e.g., 1st week, 2nd week etc... and added to appropriate place to NodeList. In either case after completion, step S1306 is executed, wherein nodes are created only for 'Days' e.g., 1, 3, 27 etc... for each of the above 'Weeks' for which data exists in

the knowledge base and are added to appropriate places in NodeList. In step S1307, Iweek is incremented and the process continues to step S1302, till Iweek is equal to the week part of CurrD, at which point the process returns.

Fig. 14 is a flowchart of the procedural steps of GetCurrWeekNodes (NodeList, CurrD) of step S1007 of this embodiment. Initially, in step S1401, Iday is set to StartofWeek(CurrD) i.e., the starting day of the current week. A check is made in step S1402 to determine if Iday is greater than the Day(CurrD) i.e., the day part of the CurrD. If so, the process returns. If not, in step S1403, a check is made to determine if Iday is equal to Day(CurrD)-1. If so, step S1404 is executed in which, a node called 'Yesterday' is created and added to appropriate place in NodeList and continues to step S1408. If not, step S1405 is executed in which, a check is made to determine if Iday is equal to Day(CurrD) i.e., current day. If so, step S1406 is executed in which, a node called 'Today' is created and added to appropriate place in NodeList and continues to step S1408. If not, nodes are created only for 'Days' e.g., 1, 3, 4 etc... for which data exists in the knowledge base and added to appropriate places in NodeList and continues to step S1408. In step S1408, child nodes are created for sessions of day(s) for which data exists in knowledge base and added to appropriate places to NodeList. Finally in step S1409, Iday is incremented by 1 and the process continues to step S1402, till Iday > Day(CurrD), at which point the process returns.

Fig. 15 is a flowchart that procedural steps of S708, ShowLinks of this embodiment. A check is made in step S1501 to determine if all the information needs to be displayed. If not, the process proceeds to step S1502, wherein based on either User settings and/or interaction and/or input and/or System settings, the

filter or the constraint on the information to be displayed is obtained. In either case the process proceeds to step S1503, wherein the relevant KPTAction and the associated KPTDocument are got from Knowledge Base. In step S1504, a check is made to determine if the Sort Item is equal to Organizations. If so, the information is sorted by Organization, as explained in detail later in Fig. 21, in step S1505 and proceeds to S1509, where it is displayed, an example of which is shown in Fig. 45. If not, a check is made in step S1506 to determine if the sorting is by Domains. If so, the information is sorted by Domain, as explained in detail later in Fig. 20, in step S1507 and proceeds to step S1509, where it is displayed, an example of which is shown in Fig. 46. If not, the information is sorted by Keywords, as explained in detail later in Fig. 16, in step S1508 and proceeds to step S1509, where it is displayed, an example of which is shown in Fig. 47 and the process returns.

Fig. 16 is a flowchart of the procedural steps of Sort by keywords S1508 of this embodiment. Initially in step S1601 the NodeList is set to NULL. In step S1602, the list of all keywords L1, is retrieved from the knowledge base. In step S1603, the next keyword K1 is fetched from the list of keywords L1. A check is made in step S1604 to determine if K1 exists. If so, a check is made in step S1606 to determine if it is a required keyword. If not, the control goes back to step S1603. If so, a check is made in step S1607 to determine if the keyword K1 already exists in the NodeList. If so, the control goes back to step S1603. If not, the keyword K1 is added at the appropriate place in the NodeList in step S1608 and control goes back to step S1603 to fetch the next keyword from the list. If K1 does not exist in step S1604, implying that all the keywords in the NodeList were

processed and hence in step S1605, ShowTreeView(NodeList, Type) is executed with Type=Keyword, whose details are explained in Fig. 17 and the example results are shown in Fig. 47, wherein the keywords in the NodeList are displayed in the sorted order.

Fig. 17 is a flowchart of the procedural steps of ShowTreeView S1605 of this embodiment. First in step S1701, a check is made to determine if Type is Keyword. If so, No keywords is added to the NodeList in step S1702. In step S1703, the list of nodes in the NodeList is displayed. In step S1704 the process waits for user operation or Action Act and in step S1705, a check is made to determine if the Act is End, in which case the process returns. If not, a check is made in step S1706 to determine if a Leaf was selected. If so ProcessLeafAction(Act, Node, Type) is executed in step S1707. If not, ProcessNodeAction(Act, Node, Type) is executed in step S1708 and the process returns to step S1704.

Fig. 18 is a flowchart of the procedural steps of ProcessLeafAction(Act, Node, Type) of step S1707 of this embodiment. A check is made in step S1801, if the Act is Open. If so, all the child nodes and all the actions KPTAction and associated KPTDocument are fetched in step S1802, from the knowledge base for the selected node and added to the NodeList at appropriate places in step S1803 and continues to step S1809. If not, a check is made in step S1804, if the Act is Close. If so, all the child nodes below the selected node are closed or hidden in step S1805 and continues to step S1809. If not, a check is made to determine if the Act is Delete. If so, a confirmation is sought from the user, if required, in step S1807 and if delete is not to be performed, it continues to step S1809, else all the

KPTAction and associated KPTDocument for all the child nodes below the selected node is deleted from the knowledge base in step S1808 and continues to step S1809. In step S1809, the knowledge base is updated based on the type of action performed and in step S1810 the user interface is updated to reflect the updates made in the knowledge base. If in step S1806, the action is not Delete, the process returns.

Fig. 19 is a flowchart of the procedural steps of ProcessNodeAction(Act, Node, Type) of step S1708 of this embodiment. A check is made in step S1901 to determine if the Act is Display i.e., to display the contents of the stored page, if contents are stored, otherwise, the original page needs to be displayed. If so, KPTAction and associated KPTDocument are fetched from the knowledge base for the selected node and added to the NodeList at appropriate place in step S1902 and continues to step S1914. If not, a check is made in step S1903 to determine if the Act is Source i.e., to display the contents of the original page. If so, the KPTAction and associated KPTDocument are fetched from the knowledge base for the selected node in step S1904 and fetches the contents of the page from the original location or URL in step S1905 and continues to step S1914. If not, a check is made to determine if the Act is Delete in step S1906. If so, a confirmation is sought from the user, if required, in step S1907 and if delete is not to be performed, it continues to step S1914, else in step S1908, the associated KPTAction and KPTDocument are deleted from the knowledge base and continues to step S1914. If not, a check is made in step S1909 to determine if the Act is Delete from this group. If so, a confirmation is sought from the user, if required, in step S1910 and if delete is not to be performed, it continues to step

S1914, else in step S1911, the associated attributes or properties of KPTAction and KPTDocument are modified in the knowledge base and continues to step S1914. If not, a check is made in step S1912 to determine if the Act is Show Property. If so, the associated properties or attributes of the KPTAction and KPTDocument for the associated node are fetched from the knowledge base in step S1913 and continues to step S1914. In step S1914, the knowledge base is updated based on the type of action performed and in step S1915 the user interface is updated to reflect the updates made in the knowledge base. If in step S1912, the action is not Show Property, the process returns.

Fig. 20 is a flowchart of the procedural steps of Sort by Domain step S1507 of this embodiment. Initially the NodeList is set to NULL in step S2001. In step S2002, all the top-level domain list L1 are fetched from the knowledge base. In step S2003, the next domain name K1 in list L1 is fetched. A check is made in step S2004 to determine if the domain name K1 exists. If so, a check is made in step S2006 to determine if the domain name K1 is unnecessary or not required domain. If so, it continues to step S2003 to fetch the next domain name from the list. If not, a check is made in step S2007 to determine if the K1 is already present in L1, if so, it continues to step S2003 otherwise the domain name K1 is added to the NodeList at the appropriate place in step S2008 and then continues to step S2003. In step S2004, if K1 does not exist, implying that all the items in the list L1 have been processed, Fig. 17 ShowTreeView(NodeList, Type), with Type = Domain is executed in step S2005 and the example results are shown in Fig. 46, wherein the Domains in the NodeList are displayed in the sorted order.

Fig. 21 is a flowchart of the procedural steps of Sort by Organization step

S1505 of this embodiment. Initially the NodeList is set to NULL in step S2101. In step S2102, list of all the organizations L1 are fetched from the knowledge base. In step S2103, the next organization name K1 in list L1 is fetched. A check is made in step S2104 to determine if the organization name K1 exists. If so, a check is made in step S2106 to determine if the organization name K1 is unnecessary or not required entry. If so, it continues to step S2103 to fetch the next organization name from the list. If not, a check is made in step S2107 to determine if the K1 is already present in L1, if so, it continues to step S2103 otherwise the organization name K1 is added to the NodeList at the appropriate place in step S2108 and then continues to step S2103. In step S2104, if K1 does not exist, implying that all the items in the list L1 have been processed, Fig. 17 ShowTreeView(NodeList, Type), with Type = Orgn is executed in step S2105 and the example results are shown in Fig. 45, wherein the Organizations in the NodeList are displayed in the sorted order.

Fig. 48 is an example for User Interface of Search UI, displayed in step S710. Fig. 22 is the flowchart for processing the retrieve conditions set in Fig. 48 and to fetch the appropriate information. In step S2201, Retrieve Query 'Q' is set to NULL. A check is made in step S2202 to determine if keyword is NULL. If not, the input keyword is set to keyword of the Retrieve Query Q in step S2203 to retrieve the matching information and continues to step S2204.If so, a check is made in step S2204 is determine if Action Type is null. If not, the input Action Type is set to Action Type of query Q in step S2205 and continues to step S2206. If so, a check is made in step S2206 to determine if Browse Date/ Validity Range is equal to NULL. If not, the input information is set to Browse Date/ Validity

Range of Retrieve Query Q in step S2207 and continues to step S2208. If so, a check is made in step S2208 to determine if URL Orgn is null. If not, the input characters of URL Orgn is set to Organization name of URL of the retrieve query Q in step S2209 to retrieve query Q to retrieve the matching information starting the given input and continues to step S2210. If so, a check is made in step S2210 to determine if URL Domain is null. If not, the input characters of URL Domain is set to Domain part of URL of the retrieve query Q in step S2211 to retrieve query Q to retrieve the matching information containing the given input and continues to step S2212. If so, a check is made in step S2212 to determine if the input Page tile is NULL. If so, the input characters of Page title is set to Page Title of the retrieve query Q in step S2213 to retrieve query Q to retrieve the matching information starting the given input and continues to step S2214. If so, a check is made in step S2214 to determine if retrieve query Q is NULL i.e., if any of the conditions to find was specified. If so, step S2215 is executed in which an error message is displayed to the user and continues to step S2202. If not, step S2216 is executed to retrieve all the matching KPTAction and associated KPTDocument from the knowledge base for the input retrieve query Q and the results are displayed to the user in step S2217 and the function returns.

Fig. 50 is an example for User Interface of Manage UI, displayed in step S712. Fig 23 is the flowchart for processing the manage conditions set in Fig. 50 and to process the appropriate request. In step S2301, Manage Query 'Q' is set to NULL. A check is made in step S2302 to determine if Content Type is NULL. If not, the input Content Type is set to Content Type of the Manage Query Q in step S2303 to retrieve the matching information and continues to step S2304. If so, a

check is made in step S2304 to determine if Browse Date is equal to NULL. If not, the input information is set to Browse Date of Manage Query Q in step S2305 and continues to step S2306. If not, a check is made in step S2306 to determine if the Delete is not based on Keywords or Organization or Domain. If so, step S2307 ShowLinks is executed, which was explained in Fig. 15 and process returns. If so, a check is made in step S2308 to determine if Manage Query Q is NULL, i.e., if any of the conditions to manage was specified. If so, step S2311 is executed in which an error message is displayed to the user and continues to step S2302. If not, step S2309 is executed to retrieve all the matching KPTAction and associated KPTDocument from the knowledge base for the input manage query Q and the results are displayed to the user in step S2310 and the function proceeds to step S2312. In step S2312, first the User Operation is obtained in Act and a check is made to determine if Act is equal to NULL. If so, the process just waits for user operation. If not, step S2313 is executed wherein a check is made to determine if the Act is Delete All. If so, all the displayed KPTAction and the associated KPTDocument are deleted from the knowledge base after confirmation from the user and proceeds back to step S2312. If not, a check is made in step S2315 to determine if the Act is equal to Delete. If so, step S2316 is executed wherein the selected KPTAction and the associated KPTDocument are deleted form the knowledge base after confirmation from the user and moves to step S2312. If not, a check is made in step S2317 to determine if Act is equal to NULL. If so, the function just returns otherwise a check is made in step S2318 to determine if Act is equal to New, in which case the input parameters are cleared and moves to step S2312.

Fig. 24 is a flowchart of the procedural steps of S619 of this embodiment, in which a check is made to see if any system task needs to be executed. In step S2401, a check is made to determine if any system action exists, which needs to be executed. If no such action exists, the process is terminated and returns. If system action exists, system task T1 is created in step S2402. A check is made in step S2403 to determine if the system action is CheckExpired. If so, RemoveExpired is executed in step S2404 to remove the information whose validity period has expired. If not, a check is made in step S2405 to determine if the system action is CheckUpdate. If so, Update is executed in step S2406 to update the version of the system. After the completion of the above steps, knowledge base is updated in step S2407 and system task T1 is terminated in step S2408 and the process returns.

Fig. 25 is a flowchart of the procedural steps of RemoveExpired S2404 of this embodiment. Referring to the figure, in step S2501, a valid or active KPTAction is fetched from the knowledge base. In step S2502, a check is made to determine if no KPTAction was found in the previous step. If so, the process terminates and returns. If not, a check is made in step S2503 to determine if the validity period of the KPTAction is defined. If not, the control goes back to step S2502 to fetch the next valid or active KPTAction from the knowledge base and the process continues. If so, a check is made in step S2504 to determine if the validity period of the KPTAction has expired. If not, the control goes back to step S2501 otherwise, a confirmation is made to the user or based on user settings without asking the user every time, to verify if the found action can be deleted, in step S2505. If so, in step S2506 the KPTAction and the associated KPTDocument

is deleted from the knowledge base and in step S2507 the knowledge base is updated and the control goes back to step S2501 to fetch the next valid or active KPTAction from the knowledge base and the process continues till no more actions are found.

Fig. 26 is a flowchart of the procedural steps of CheckUpdate S2406 of this embodiment. In step S2601, the current version of the system, V1 is obtained. In step S2602 a check is made to determine if the version V2 can be obtained from the website. If not, the process terminates and returns. If so, a check is made to determine if version V2 is greater than V1. If not, the process terminates and returns. If so, a check is made in step S2604, to see if user confirmation is required, in which case after confirming from the user, the latest version is downloaded in step S2605 and the system updates itself in step S2606 and the process returns.

Fig. 27 is a flowchart of the procedural steps of S622, ExecuteAction of this embodiment. In step S2701, the next Act is got from the ActList. In step S2702, a check is made to determine if Act exists. If not, the process returns. Otherwise, in step S2703 inference is made using the knowledge base to complete the Act. A check is made in step S2704 to determine if Act is Save. If so, step S2705, SaveContents() as explained later in Fig. 28 is executed and goes to step S2706 wherein the knowledge base is updated and the process returns to step S2701 to fetch the next action from the ActList, till there are no more action left to be processed, at which stage the process returns.

Fig. 28 is a flowchart of the procedural steps of SaveContents in step S2705 of this embodiment. A check is made in step S2801 to determine if it is a

SaveLink only operation. If so, process proceeds to step S2805. Otherwise, a check is made to determine if it is a SavePage contents operation in step S2802. If so, Page PLUS is set to true in step S2804. In either case, step S2803, WebFetch() is executed, which is explained in detail later in Fig. 29, in step S2803. In step S2805, a check is made to determine if ModifyStatus is saveAsNewAction or not. If so, indices of KPTAction and the associated KPTDocument is determined from Knowledge Base in step S2809 and SaveFileContents() is executed as explained in Fig.30, in step S2801. The KPTAction and KPTPerson are added to Knowledge Base in step S2806 and the process returns. If ModifyStatus is not saveAsNewAction, check is made in step S2807 to determine if it is OverWriteExisiting. If not the process returns, otherwise, in step S2811 indices of KPTAction and the associated KPTDocument is determined from Knowledge Base in step S2811 and SaveFileContents() is executed as explained in Fig.30, in step S2812. The KPTAction and KPTPerson are updated in the Knowledge Base in step S2808 and the process returns.

Fig. 29 is a flowchart of the procedural steps of WebFetch in step S2803 of this embodiment. In step S2901, HTML document obtained from the browser is opened. In step S2902, next tag is got. In step S2903, a check is made to determine if the end of file has been reached. If so the process returns. If not, a check is made to determine if the tag is for an embedded image, frame etc. in step S2904. If so, step S2905 is executed. If not, a check is made in step S2909 to determine if PagePLUS is true and the Tag type is of LINK. If not the process returns back to step S2902 to fetch the next tag. Otherwise, step S2905 is executed in which a check is made to see if the contents i.e., embedded images etc. already

exist in our knowledge base and they are up to date in step S2905. If so, the HTML tag is edited in step S2906 to change the absolute or original path to the local path of the system where the file exists and process returns to step S2902. If not, a check is made to determine if the file to be fetched is a local file in step S2910. If so, the file contents are just copied, using a simple file copy command in step S2911, otherwise the contents are downloaded from the internet in step S2907. In either case step S2908 is executed, wherein the knowledge base is modified to update the information downloaded etc. and process returns to step S2902 to fetch the next tag in the HTML document. The process continues till end of file is reached at which instant the process returns.

Fig. 30 is a flowchart of the procedural steps of steps S2810, S2812 SaveFileContents() of this embodiment. A check is made in step S3001 to determine if the contents to be saved is SaveLink only. If so, the process continues to step S3006. In step S3002, a folder F1 with the name based on the KPTDocument's name, which is a Globally unique identifier (GUID) is created, which ensures that the folder to be created is unique within and across the local system. In step S3003, a file called KPTIndex is created in the folder created in previous step. The actual page contents i.e., HTML text are saved to the file created in the previous step. The fully qualified file name i.e., the folder name and the file name are stored as the physical URL location of the KPTDocument. In step S3006, FillKPTAction() is executed which is explained in detail in Fig. 31 and the other required indices are determined by referring to the knowledge base in step S3007 and the process returns.

Fig. 31 is a flowchart of the procedural steps of S3006, FillKPTAction()

of this embodiment. In step S3101, the contents of 'URL' are set to 'LogicalURL' field of KPTDocument. In step S3102, the contents of 'keyword' are set to 'Keyword' field of KPTDocument. In step S3103, the time and date are set to 'WhenDone' field of KPTAction. In step S3104, the 'Validity' is set to 'WhenToDo' field of KPTAction and in step S3105, 'Page title' is set to 'Title' of KPTDocument and process returns.

Thus, when the user just clicks on Save Page or Save link, this embodiment does not ask the user for file name or destination folder location and the actual process of saving the information in a one-touch operation. Also the same information be it a page or link can be stored multiple times i.e., store the contents of the same link as many times as the user wishes, without ever specifying the filename, and still be able to retrieve them easily. This is especially useful, for users to keep track of the changes in the contents of the same web page.

Fig. 32 is a flowchart of the procedural steps of TERMINATE of step S403 of this embodiment. In step S3201, the UI on display are closed. In step S3202, all the current sessions are ended. In step S3203, Knowledge base is updated. A check is made in step S3204 to determine if browser needs to be ended or terminated. If so, the browser will be terminated in step S3205 and the process ends.

Fig. 33 shows an example of the knowledge structures in the knowledge base. (a), (b), (c) are the knowledge structure definitions for KPTConcept, KPTPerson and KPTDocument respectively.

Fig. 34 shows an example of the knowledge structures in the knowledge

base. (a), (b) are the knowledge structure definitions for KPTAction and KPTContent respectively.

Fig. 35 shows an example content of the knowledge base. (a), (b) are the contents of the knowledge base for KPTDocument and KPTAction respectively.

Fig. 36 shows an example content of the knowledge base. (a), (b) are the contents of the knowledge base for KPTPerson and KPTContent respectively.

Fig. 37 shows an example content of the Domain knowledge base.

Fig. 38 shows example rules for determining the organization indices for a URL. As can be seen from the figure, the parameter, protocol, obvious address, page information is first removed from the URL. By referring to the Domain Knowledge base, whose example is shown in Fig. 37, the domain part of the URL is removed. The remaining URL is then processed to obtain the Organization indices as shown.

Fig. 39 shows example rules for determining the domain indices for a URL. As can be seen from the figure, the parameter, protocol, obvious address, page information is first removed from the URL. By referring to the Domain Knowledge base, whose example is shown in Fig. 37, the domain part of the URL is obtained.

Fig. 40 shows an example User Interface of this embodiment. The keywords obtained in step S612 – Saora, Web Information Manager etc. are displayed in the UI as shown in step S618.

After browsing on the Internet, it often happens that one would like not only to revisit the pages one had seen earlier but also view the associated links in the same order one had previously visited during that session. This embodiment

facilitates this type of tracking. When the user feels that he might need to keep track of the pages they are browsing in the current session, they can just use the Save Link, to save only the URL(s) or links. Later, by using the Show Sessions functionality as explained in Fig. 9, they will be able to view, revisit or replay the links in the order they had visited them earlier. Thus by just clicking on the links they will be able to recreate the previously browsed session.

If the Save Page option is used instead of Save Link, they will be able to track or recreate the session offline i.e., without connecting to the Internet to fetch the pages again or worry about broken links or Page not found errors.

Fig. 41 shows an example HTML text obtained from the browser. In step S621, the system obtains the HTML text like shown in this figure and in steps S2904 to S2908, downloads the embedded files, which happens to be image files with '.gif' extension for the example HTML file shown.

Fig. 42 shows example modified HTML text. The original HTML text obtained from the browser, shown in Fig. 41 is modified by the system in step S2906 and the resultant HTML text is shown in this figure. As can be seen from this figure, the embedded links are modified to reflect the path where the system stores the embedded contents.

Fig. 43 shows an example User Interface of this embodiment when a page already stored is retrieved. As can be seen from the figure, being a Retrieved URL, the RetrieveUI message is obtained in step S617 and displayed in S618.

As can be seen from this figure, the saved web pages can be retrieved WYSIWYG (what you see is what you get) including embedded images, frames etc... which allows for offline browsing, thus eliminating the need to be

connected to the Internet. This embodiment enables the user, when short of time to keep the web information and view it at leisure offline. The stored contents can be viewed offline using Show Sessions or Get All Links, allowing them to carry it on their mobile computer, thus eliminating the printouts for out-of-office or home reading.

Fig. 44 shows an example User Interface of Session UI. In step S904, for each session the KPTAction is sorted by Time and the result is as shown in this figure.

Fig. 45 shows an example User Interface of Get All Links UI. In step S1505, the links are sorted by Organization and the resultant UI displayed in step S1509 is as shown. As can be seen from the figure, if a content has subdomains i.e., sales.saora.com then it will appear in multiple places below each of the organization nodes i.e., sales, Saora etc....

Fig. 46 shows an example User Interface of Get All Links UI. In step S1507, the links are sorted by Domains and the resultant UI displayed in step S1509 is as shown.

Fig. 47 shows an example User Interface of Get All Links UI. In step S1508, the links are sorted by Keywords and the resultant UI displayed in step S1509 is as shown. As can be seen from the figure, if a content has multiple keywords then it will appear in multiple places below each of the keyword nodes.

Fig. 48 shows an example Retrieve or Find UI, which is displayed in step S710.

Fig. 49 shows an example Properties or Attributes UI of an entry selected in Fig. 48.

Fig. 50 shows an example Manage or Delete Pages or Links UI, which is displayed in step S712.

Figs. 50 to 53 show some examples of User Settings UI, displayed in step S714 of this embodiment.

The present invention described above may be applied to a system constituted of a plurality of computers, or a specific computer within a system. the object of the present invention can also be achieved by supplying a storage medium storing program codes of software for implementing the function of the above embodiment to a system or an apparatus, and reading out and executing the program codes stored in the storage medium by a computer (or a CPU or MPU) of the system or apparatus. In this case, the program codes read out from the storage medium implement the function of the present invention, and the storage medium storing these program codes constitutes the invention. Also, besides the function of the above embodiment is implemented by executing the readout program codes by the computer, the present invention includes a case where an OS (Operating System) or the like running on the computer performs a part or the whole of actual processing in accordance with designations by the program codes and thereby implements the function of the above embodiment.

Furthermore, the present invention also includes a case where, after the program codes read out from the storage medium are written in a memory of a function extension board inserted into the computer or of a function extension unit connected to the computer, a CPU or the like of the function extension board or function extension unit performs a part or the whole of actual processing in accordance with designations by the program codes and thereby implements the

function of the above embodiment.

Although the present invention has been described in its preferred form with a certain degree of particularity, many apparently widely different embodiments of the invention can be made without departing from the spirit and scope thereof. It is to be understood that the invention is not limited to the specific embodiments thereof except as defined in the appended claims.